Filed on behalf of: R2 Solutions LLC

By: Carder W. Brooks
Registration No. 75,456
NELSON BUMGARDNER CONROY P.C.
3131 W. 7th St., Suite 3000
Fort Worth, TX 76107
Telephone: (817) 806-3814
Email: carder@nelbum.com

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CLOUDERA, INC.,

Petitioner

v.

R2 SOLUTIONS LLC,

Patent Owner.

Case IPR2024-00303

U.S. Patent 8,190,610

PATENT OWNER'S PRELIMINARY RESPONSE

TABLE OF CONTENTS

I. INTRODUCTION 1

II. BACKGROUND 1

A. Overview of the ’610 Patent 1

1. Introduction to MapReduce 2

2. Shortcomings of Prior Technologies 5

3. Improvement to MapReduce Described by ’610 Patent 8

4. The Claimed Technology of the ’610 Patent Improves MapReduce. 12

III. LEVEL OF ORDINARY SKILL 14

IV. CLAIM CONSTRUCTION 15

V. APPLICABLE LEGAL STANDARDS 18

A. Obviousness 18

1. Claims Cannot Be Found Obvious if an Element Is Absent. 19

2. A Petition Must Provide Articulated Reasoning with Rational Underpinning to Combine and/or Modify References. 20

VI. SUMMARY OF THE PRIOR ART REFERENCES 21

A. Overview of Primary Reference: Pike 21

B. Overview of Secondary References: Chowdhuri 24

VII. THE PETITION SHOULD BE DENIED UNDER 35 U.S.C. § 325(d) 27

A. Substantially the Same Art and Arguments Were Previously Presented to the Office. 28

B. Petitioner Has Failed to Demonstrate that the Office Committed Material Error. 32

VIII. THE PETITION DOES NOT DEMONSTRATE THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE. 34

A. Ground 1: Pike Does Not Teach or Suggest at Least One Limitation of Each Independent Claim. 36

1. Pike Does not Disclose "Data Groups." 36

2. Pike Does not Disclose Delineating "Data Groups" Prior to Partitioning. 40

3. Pike Does not Disclose "Intermediate Data" that Is "Identifiable" to a "Data Group." 44

4. Pike Does not Disclose "Data Groups" with Different Schemas. 48

5. Pike Does not Disclose a Key in Common Between the (1) "Data Groups"/Schema of the Input Data, and (2) the Intermediate Data.....51

6. Pike Does not Disclose Processing Intermediate Data According to its "Data Group," or Merging the Intermediate Data Based on a Key in Common Between the Intermediate Data and the "Data Groups."...53

B. Grounds 2 and 3: Combining Chowdhuri with Pike Fails to Render the Independent Claims Obvious. 56

1. Chowdhuri Does not Disclose "Data Groups." 56

2. Chowdhuri Does not Disclose Delineating "Data Groups" Prior to Partitioning. 60

3. Chowdhuri Does not Disclose "Providing Each Data Partition to a Selected One of a Plurality of Mapping Functions." 61

4. Chowdhuri Does not Disclose "Intermediate Data" that Is "Identifiable" to a "Data Group." 62

5. Chowdhuri Does not Disclose "Data Groups" with Different Schemas. 65

6. Chowdhuri Does not Disclose a Key in Common Between the (1) "Data Groups"/Schema of the Input Data, and (2) the Intermediate Data. 66

7. Chowdhuri Does not Disclose Processing Intermediate Data According to its "Data Group," or Merging the Intermediate Data Based on a Key in Common Between the Intermediate Data and the "Data Groups." 67

IX. CONCLUSION.....68

TABLE OF AUTHORITIES

Cases

Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH,
IPR2019-01469 (PTAB Feb. 13, 2020) ..... *passim*

American Airlines, Inc. et al. v. R2 Solutions LLC,
IPR2023-00689 (PTAB Mar. 7, 2023) 34

CFMT, Inc. v. YieldUp Int'l Corp.,
349 F.3d 1333 (Fed. Cir. 2003) 19

Garmin Int'l, Inc. v. Patent of Cuozzo Speed Techs. LLC,
Case No. IPR2012-00001, Paper 15 (PTAB Jan. 9, 2013) 19

Graham v. John Deere Co.,
383 U.S. 1 (1966) 18

In re Kahn,
441 F.3d 977 (Fed. Cir. 2006) 20

In re Rijckaert,
9 F.3d 1531 (Fed. Cir. 1993) 19

In re Royka,
490 F.2d 981 (C.C.P.A. 1974) 19

InTouch Techs., Inc. v. VGo Comm'ns., Inc.,
751 F.3d 1327 (Fed. Cir. 2014) 19

KSR Int'l Co. v. Teleflex Inc.,
550 U.S. 398 (2007) 19, 20, 66

LG Elecs., Inc. v. Cellular Commc'ns Equip. LLC,
Case No. IPR2016-00197, Paper 7 (PTAB April 29, 2016) 20, 67

N.V. v. Abbott Labs.,
512 F.3d 1363 (Fed. Cir. 2008) 20

Rules, Regulations, and Statutes

35 U.S.C. § 103 19

35 U.S.C. § 103(a) 18

35 U.S.C. § 316(e) 18

35 U.S.C. § 325(d) 1, 27, 34

37 C.F.R. § 42.108(c).....18

Other Authorities

PTAB Consolidated Trial Practice Guide (Nov. 2019) 18

PATENT OWNER'S EXHIBIT LIST

Exhibit	Description
2001	[RESERVED]
2002	Fundamentals of MapReduce
2003	MapReduce Design Patterns
2004	[RESERVED]
2005	Google's MapReduce patent – what does it mean for Hadoop?
2006	Information Disclosure Statement
2007	US6341289
2008	US20060117036A1
2009	"Hash join in MySQL 8," *available at* https://dev.mysql.com/blog-archive/hash-join-in-mysql-8/.

I. INTRODUCTION

The Petition should be denied under 35 U.S.C. § 325(d). The Petition presents substantially the same art and arguments that were already before the USPTO during prosecution of the ’610 patent. Thus, under *Advanced Bionics*, it was incumbent on Petitioner to demonstrate how the USPTO erred in a way material to patentability. *See Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 at 7 (PTAB Feb. 13, 2020) (precedential). But the Petition is silent as to any alleged error committed by the USPTO. The Board should thus exercise its discretion under § 325(d) and deny institution.

The Petition should also be denied under 35 U.S.C. § 314(a) because it fails to establish a reasonable likelihood that the Board would find any of the challenged claims unpatentable. The ’610 patent claims novel and non-obvious enhancements to MapReduce architecture that are nowhere to be found in the prior art and would not be obvious to a POSITA. Each of the Petition’s grounds have glaring holes, neither of the cited references teach or suggest key limitations of the challenged claims, and the combination of the references is self-evidently inoperable.

II. BACKGROUND

A. Overview of the ’610 Patent

The ’610 patent, granted to Yahoo!, claims a novel MapReduce architecture that can be applied to affect mapping and reducing across data sets comprising data

of different schema. Such architecture, adapted to receive and process data from heterogenous sources, greatly expanded the applicability of MapReduce to dramatically improve big data analytics. This improvement was significant at the time of the ’610 patent (and remains significant today), enhancing the overall capabilities of computer functionality as it relates to distributed database processing.

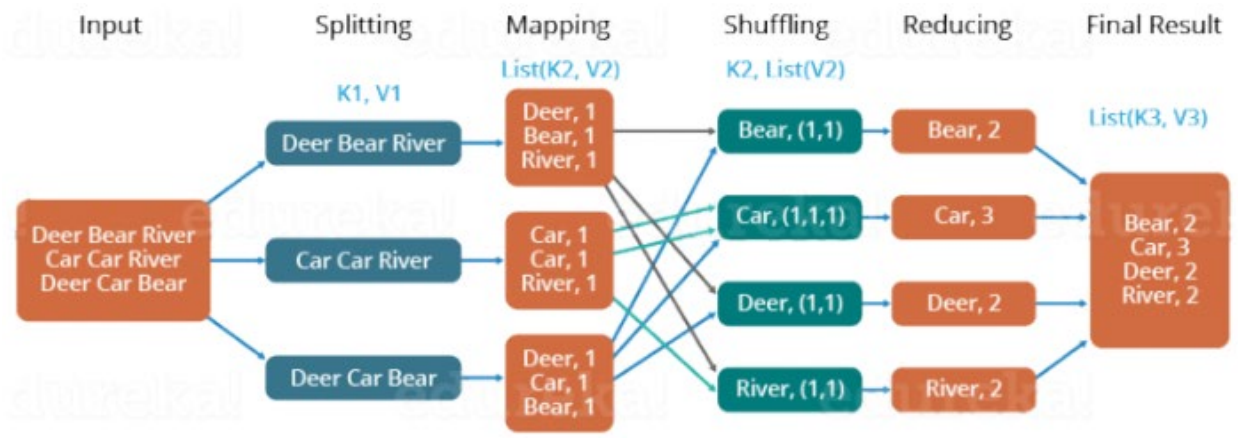
1. Introduction to MapReduce

As laid out in the ’610 specification, the concept of MapReduce consists of a “‘map’ function [that] maps key-value pairs to new (intermediate) key-value pairs,” and a “‘reduce’ function [that] represents all mapped (intermediate) key-value pairs sharing the same key to a single key-value pair or a list of values.” ’610 patent at 1:17-20. At its core, “MapReduce implementations enable the use of massive clusters of commodity computers to provide a simplified programming and execution model for processing large sets of data in parallel.” *Id*. at 9-12.

In a simple example found in U.S. Patent No. 7,590,620 to Pike (“Pike,” Petitioner’s primary reference), MapReduce can be used to “count[] the number of occurrences of each distinct word in a large collection of documents.” Ex. 1002 at 10:36-39. In this example, the map function “outputs a key/value pair for every word w in every document in the collection, where the key/value pair is <w, 1>.” *Id*. at 10:40-43. It is likely that with a voluminous number of documents, each map function “will produce hundreds or thousands of records of the form <word, 1>.” *Id*.

at 10:51-52. After the mapping phase, the reduce functions "sort the intermediate data values, merge or otherwise combine sorted intermediate data values having the same key." *Id*. at 8:16-19. In this example, the effect is that "[t]he Reduce function simply adds up the count values." *Id*. at 10:52-53.

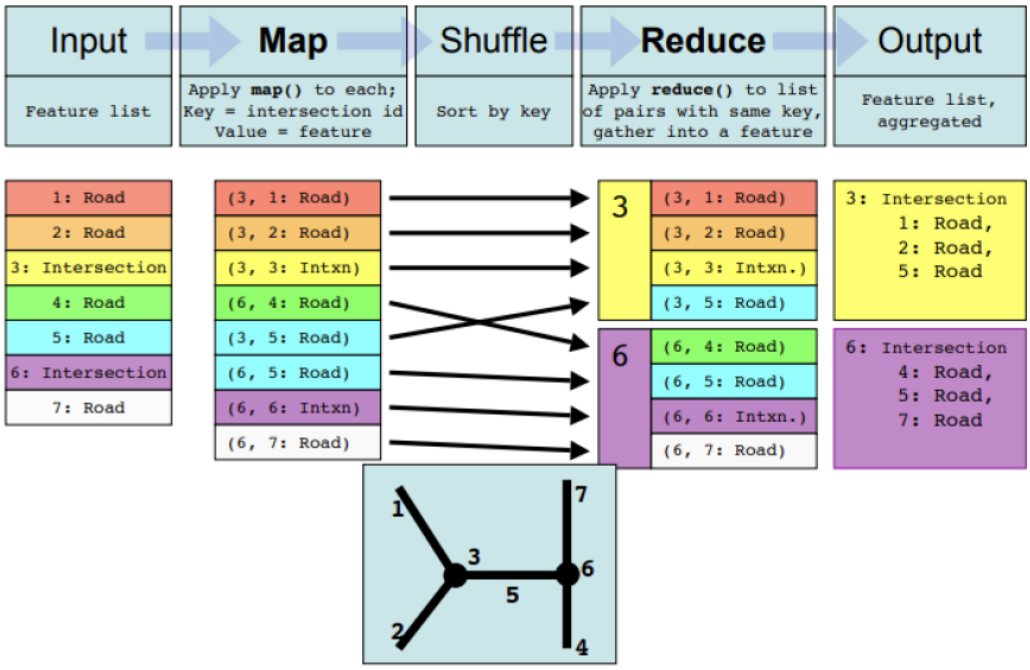
The below graphic illustrates the word count example discussed in Pike:



Ex. 2002 at 5. In this example, the input data consists of a collection of words (Deer, Bear, River, and Car). Applying traditional MapReduce in the "Splitting" step, the input data is split to be supplied to different map processes. Next, in the "Mapping" step, each map function creates lists of key-value pairs. A key-value pair works much like a dictionary—the words are the "keys," and the definitions are the "values." In the example above, each mapping function generates a list of key-value pairs where the words are the "keys" and the values are "1" to indicate that each word, as a discrete item, occurs one time (e.g., "Deer, 1"). *See* Ex. 1002 at 10:40-43. Obviously, some of the words occur multiple times, and that is accounted for later in the process.

At the "Shuffling" step, all key-value pairs having the same key are grouped together to ensure that pairs with the same key are submitted to the same reducers. At the "Reducing" step, the reducers determine how many occurrences of that particular key exist for the received data and can generate another key-value pair, accordingly. *See id.* at 8:16-19. Ultimately, the output in this example is a list of all of the key-value pairs from each reducer that represents the number of times the word appears in the input text (e.g., "Deer, 2" indicating that the word "Deer" appeared twice). *See id.* at 10:52-53; 8:16-19.

MapReduce can also be used to accomplish other tasks, such as "joins" or "merges" of data or lists that already exist as key/value pairs. *See id*. at 9:65-66 ("The input data blocks . . . may in some embodiments be treated as key/value pairs . . ."). The below graphic illustrates such a merger:



Ex. 2003 at 8.

Here, the input data is already provided as a list of key-value pairs. In this example, the input data is a list of features in a graph (e.g., roads and intersections) that are numbered; the numbers are the keys and the feature types are the values. In the mapping phase, new keys are assigned that correspond to the intersection that the particular feature participates in (e.g., roads 1, 2, and 5, and intersection 3, all are involved with intersection 3). *See* Ex. 1002 at 7:29-43. Thus, in this example, the key generated in the mapping phase is the intersection ID, and the value is the key-value pair corresponding to the graph feature. The values are shuffled and passed to the reducers based on their common key. *See id*. at 8:16-19. Then, the reducers "reduce" the data to yield the output data, i.e., the reducers can consolidate all of the features pertaining to a particular common key, such that all of the features participating in intersection 3 fall with intersection 3, and all of the features participating in intersection 6 fall with intersection 6. *See id*.

As is apparent from the above examples, MapReduce offers a way to take large, unwieldy datasets and extract quantifiable and useful information algorithmically. As discussed in the '610 patent, however, in 2006, MapReduce had significant limitations that restricted its usefulness.

2. Shortcomings of Prior Technologies

The '610 patent discloses shortcomings in the prior art and then explains, in

detail, the technical way the claimed inventions resolve or overcome those shortcomings. The specification discusses that "conventional MapReduce implementations do not have facility to efficiently process data from heterogeneous sources" and that "it is impractical to perform joins over two relational tables that have different schemas." '610 patent at 3:9-20. "Schema" can refer to, e.g., "fields" in a particular collection of data. For example, in Figure 3 of the '610 patent, the "All Emp" table (302) and "All Dept" table (304) have different "schema" because the "All Emp" table has the fields "DeptID" and "EmpName," while the "All Dept" table has the fields "DeptID" and "DeptName":

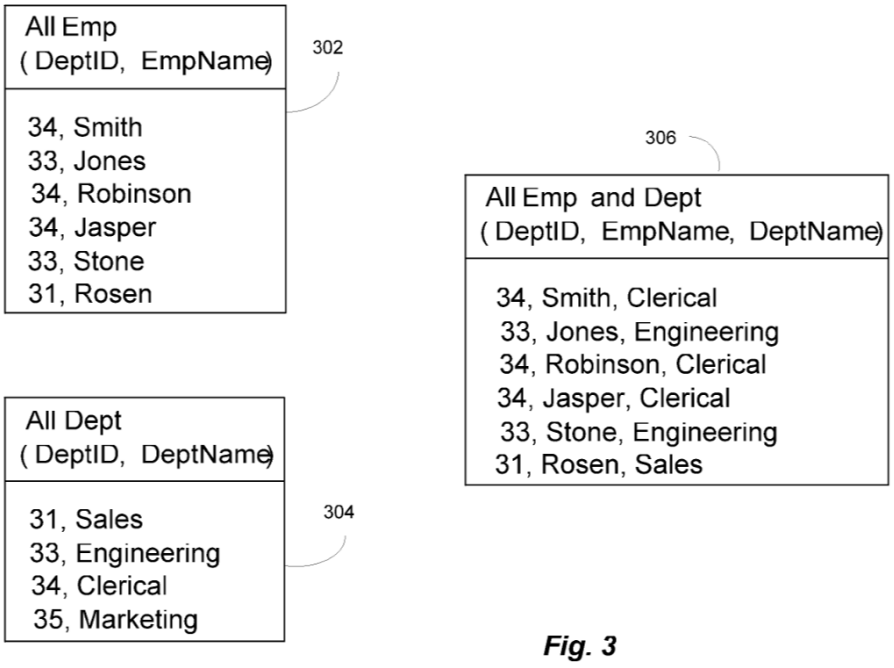
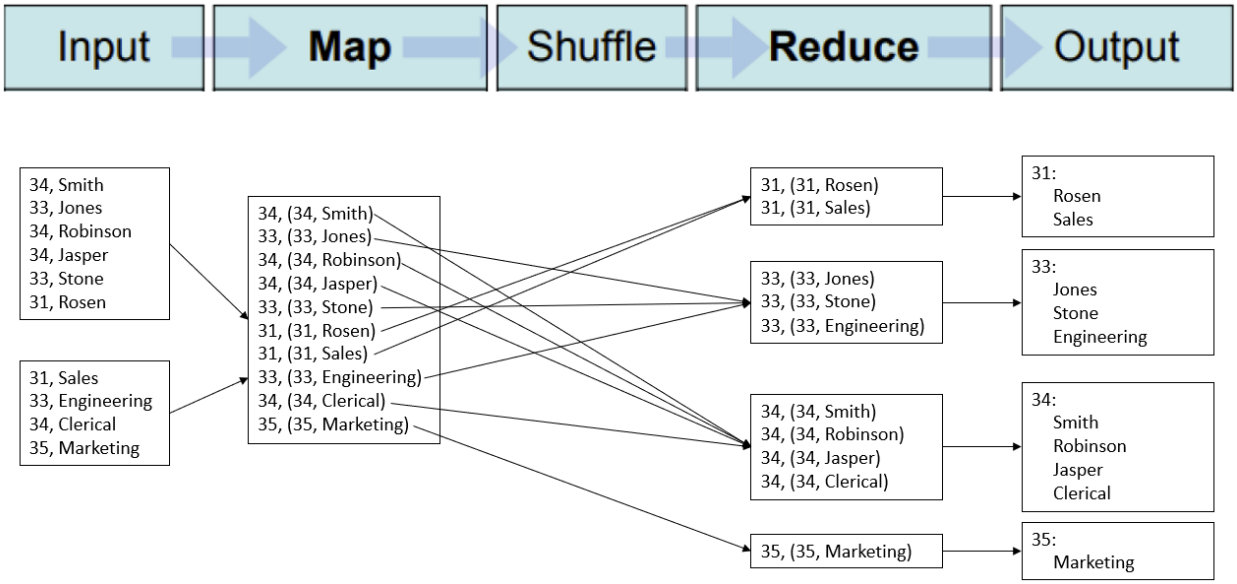


Fig. 3

Id. at FIG. 3; 3:19-20.

The specification explains that traditional MapReduce cannot efficiently join the "All Emp" table (302) and "All Dept" table (304) to achieve the "All Emp and Dept" table (306). For example, the below graphic demonstrates how traditional MapReduce would process the "All Emp" and "All Dept" tables:



In this example, the "key" for the mapping phase is the "DeptID" (which, mirroring the merger example in Ex. 2003, is a field that each entry shares or participates in) followed by a value of "DeptID, EmpName" or "DeptID, DeptName" (again, mirroring the merger example above). As can be seen here, even though there is a key-in-common amongst the "All Emp" and "All Dept" data sets at the input, there is no joining of the data in any usable way. Instead, the output is a collection of jumbled lists that show both employee names and department names for a single key instead of indicating, e.g., DeptID, employee name, and the associated department name. For instance, the output list for key 34 in this example

contains "Clerical" (a department name) mixed with "Smith," "Robinson," and "Jasper" (employee names)—there is no merger of the two tables to reflect, e.g., "34, Smith, Clerical." As datasets get bigger and more complex, this problem would only be compounded.

3. Improvement to MapReduce Described by '610 Patent.

The '610 patent offers an improvement over traditional MapReduce through the use of "data groups." The specification summarizes this enhancement:

> In accordance with an aspect, an input data set is treated as a plurality of grouped sets of key/value pairs, which enhances the utility of the MapReduce programming methodology. Utilizing such grouping, map processing is carried out independently on two or more related datasets (e.g., related by each being characterized by a schema with a key in common). The intermediate results of the map processing (key/value pairs) for a particular key are processed together in a single reduce function by applying a different iterator to intermediate values for each group. Different iterators can be composed inside reduce functions in ways however desired.
>
> Thus, for example, the enhanced MapReduce programming methodology may be easily employed to carry out distributed relational database processing.

'610 patent at 1:31-44.

The specification goes on to explain how implementation of "data groups," realizes these improvements:

> In general, partitioning the data sets into data groups enables a mechanism to associate (group) identifiers with data sets, map functions and iterators (useable within reduce functions to access intermediate data) and, also, to produce output data sets with (group) identifiers. It is noted that the output group identifiers may differ from the input/intermediate group identifiers.

Id. at 3:58-64.

The ’610 patent provides a helpful schematic that illustrates how the “data groups” improve MapReduce:

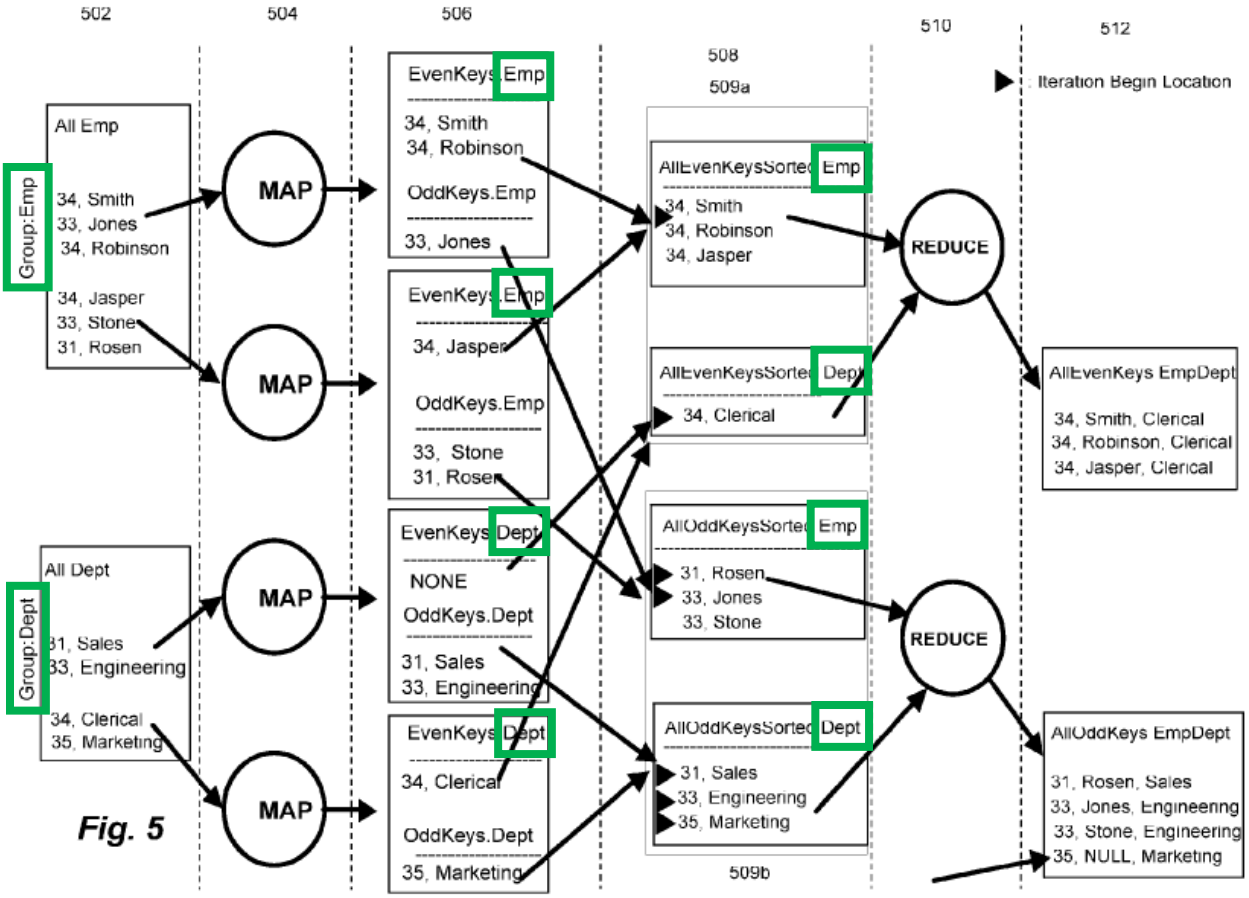


Fig. 5

Id. at FIG. 5 (annotated).

In the above schematic, the input data in portion 502 consists of the Employee table 302 and the Department table 304 from Figure 3, which share a common key (i.e., DeptID). *See id*. at 8:15-24. Each of these "data groups" include a "mechanism for identifying data from that group"[1] that is depicted as following the data through the mapping phase 504 and into the reducing phase 510 (i.e., "Emp" and "Dept" emphasized in green). As shown here, the data of each of the "data groups" are partitioned into a "plurality of data partitions" in between portions 502 and 504, and each partition is provided to a "selected one of a plurality of mapping functions" in portion 504. The mapping functions form "intermediate data" at portion 506, and the "intermediate data" is shown as being "identifiable" to its parent "data group" via the "Emp" and "Dept" mechanisms. The "intermediate data" also shares the common key (DeptID) with the two "data groups." After sorting in portion 508, "intermediate data" corresponding to each "data group" is provided to "reduce functions within the portion 510." *Id*. at 8:34-37. By "processing the intermediate

---

[1] As discussed in more detail below, the Eastern District of Texas construed the term "data group" to mean "a group of data and a mechanism for identifying data from that group." Petitioner purports to apply this construction. *See* Petition at 10. The Board should likewise adopt that construction.

data for each data group in a manner that is defined to correspond to that data group" in the reducing phase, portion 512 depicts data that is "a merging of the corresponding different intermediate data based on the key in common" (i.e., table 306 in Figure 3).

Thus, the '610 patent provides a clear technological improvement to existing MapReduce systems via a novel MapReduce architecture where mapping and reducing functions can be applied to data from heterogeneous data sources (i.e., data sources having different schema) to accomplish the merger of heterogeneous data based on a key in common among the heterogeneous data. The specification goes so far as to provide specific examples of how a "data group"-oriented MapReduce architecture can be deployed in programming. The '610 patent shows, for example, that a "data group" identifier can be passed as a parameter to a mapping function and thereby travel through to the resulting intermediate data, followed by a reduce function that employs the "data group" identifier to call particular programming objects (e.g., iterators) associated with specific "data groups":

```
map(string group, string key, string val) {
    // group: "emp" or "dept"
    // key: DeptID
    // val: LastName for "emp", DeptName for "dept"
    DeptID = key;
    emit_to_group(group, DeptID, val);
}
```

```
reduce(hashtable iterators) {
    // iterators: one iterator for "emp", another for "dept"
    DeptID = iterators.get_key( );
    emp_iter = iterators["emp"].get_iter( );
    dept_iter = iterators["dept"].get_iter( );
    for each LastName in emp_iter do {
        for each DeptName in dept_iter do {
            emit_to_group("emp_dept", DeptID, (LastName,
            DeptName));
        }
    }
}
```

Id. at 6:29-33, 39-48. Figures 3-5 illustrate a similar embodiment. In this embodiment, the "data group" identifier (annotated in red) is indicated to the mapping and/or reduce functions, such that different "data groups" can be mapped differently and/or processed in a way that corresponds to a particular "data group":

map("emp", 34, "Smith")	map("dept", 34, "Clerical")

reduce((34, ("emp", ("Smith", "Robinson", "Jasper")), ("dept", ("Clerical"))))

Id. at 7:45, 55, 64-65.

4. The Claimed Technology of the '610 Patent Improves MapReduce.

The independent claims of the '610 patent claim the improved MapReduce architecture outlined in the specification. A crucial feature is the implementation of what is referred to as a "data group." As explained above, by utilizing "data groups," a data set with data of different schema can be fed to a collection of mapping functions to ultimately be reduced and/or merged because the "mechanism for identifying data from that group" (which is part of the "data group") follows the data through the mapping phase and into the reducing phase. Thus, the MapReduce architecture can use specialized processing based on the "data group" from which the data being reduced originated. These claimed features enable the MapReduce architecture in the reducing phase to accomplish the merger of intermediate data hailing from different data groups.

For example, Claim 1 describes that "data groups," each with a different schema, are partitioned, and the resulting partitions are provided to "a selected one of a plurality of mapping functions" such that "the data of the first data group is mapped differently than the data of the second data group." Ex. 1026 at 1. This requires that the architecture differentiates between "data groups" at the mapping phase to obtain "intermediate data" on a per-"data group" basis, which avoids jumbling of data having different schemas into the same map and thus eliminating the usefulness of the reducing phase. After partitions of each of the "data groups" are provided to selected mapping functions, the claim describes that the mapping functions output "intermediate data" for a particular "data group" and that the "different schema [of the data groups] and corresponding different intermediate data have a key in common." *Id*. This "key in common" requirement is necessary, because it ultimately serves as the key on which the data of the "data groups" is merged in the reducing phase.

Claim 1 further requires that the "intermediate data" be "identifiable" to its parent "data group" (i.e., via the "mechanism for identifying data from that [data] group"). *Id*. As revealed later in the claim, the MapReduce architecture "process[es] the intermediate data for each data group in a manner that is defined to correspond to that data group, so as to result in a merging of the corresponding different intermediate data based on the key in common." *Id*. In other words, the reduce

functions recognize that received "intermediate data" stems from particular "data groups," and can therefore employ processing that corresponds to each "data group" to accomplish a combined reducing phase that merges the data (each with a different schema) together. The specification provides helpful context regarding this step, disclosing that, in one embodiment, the reducing phase can employ "data group"-specific iterators to accomplish merger, and this is only possible because of the improved MapReduce architecture's implementation of "data groups." *See*, *e.g.*, '610 patent at 6:35-7:23. Without "intermediate data" that is identifiable to a particular "data group," Claim 1's MapReduce architecture would not be able to distinguish one type of "intermediate data" from another, which would (in one example) lead to the useless amalgamation of data discussed in Section II.A.2 above.

Hence, the claimed features describe a MapReduce architecture that accounts for different "data group" schema to accomplish a reduction and merger of data having different schema. As explained in the specification, such task was ineffective using traditional MapReduce architectures. *Id*. at 3:9-20

III. LEVEL OF ORDINARY SKILL

For the limited purpose of this Preliminary Response, Patent Owner does not contest Petitioner's definition of a person of ordinary skill in the art, but it reserves the right to do so in the event that trial is instituted.

IV. CLAIM CONSTRUCTION

The Eastern District of Texas has twice construed terms of the ’610 patent, ordering the following constructions (*see* Ex. 1007 at 13-31; Ex. 1008 at 8-19):

Term/Phrase	**Agreed Construction**
“a plurality of mapping functions that are each user-configurable”	“two or more mapping functions that are each configurable by a user”
“data group”	“a group of data and a mechanism for identifying data from that group”
“a method of processing data of a data set over a distributed system, wherein the data set comprises a plurality of data groups, the method comprising”	Limiting preamble in Claim 1
“identifiable to [that/the] data group”	plain meaning

"independently output a plurality of lists of values"	plain meaning
"the data of the first data group is mapped differently than the data of the second data group"	plain meaning
"data set"	plain meaning
"data partition"	plain meaning
"including processing the intermediate data for each data group in a manner that is defined to correspond to that data group, so as to result in a merging of the corresponding different intermediate data based on the key in common"	plain meaning
"the at least one output data group is a plurality of output data groups"	plain meaning

R2 and Cloudera have not yet engaged in claim construction proceedings in their dispute in the Western District of Texas.

For the purposes of this Preliminary Response, Patent Owner submits that the Board should adopt and apply the Eastern District of Texas's construction of "data group" as "a group of data and a mechanism for identifying data from that group. Petitioner itself purports to apply this construction.[2] *See* Petition at 10. And this construction of "data groups" is proper in light of the patent itself. As explained by the court, the specification discloses that partitioning data "into data groups enables a mechanism to associate (group) identifiers" with the data ('610 patent at 3:58-59), and thus, "the word 'group' inherently implies some mechanism to identify the group. . . . Finding otherwise would potentially leave the scope of 'data group' amorphous, and some construction of the disputed claim language will assist" in understanding the claims. Ex. 1007 at 24-25 (internal citations omitted). The Board should accordingly construe "data group" to mean "a group of data and a mechanism for identifying data from that group."

---

[2] Petitioner also purports to apply the construction of "a plurality of mapping functions that are each user-configurable." This claim term is not implicated in this Preliminary Response.

V. APPLICABLE LEGAL STANDARDS

The Board may only grant a petition for *inter partes* review where "the information presented in the petition … shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." 35 U.S.C. § 314(a); 37 C.F.R. § 42.108(c). Petitioner bears the burden of showing that this statutory threshold has been met. *See* 35 U.S.C. § 316(e); *see also* PTAB Consolidated Trial Practice Guide at 3 (Nov. 2019) ("The Board … may institute a trial where the petitioner establishes that the standards for instituting the requested trial are met ….").

A. Obviousness

Section 103 of the Patent Act provides that "[a] patent may not be obtained . . . if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains." 35 U.S.C. § 103(a). The obviousness analysis requires several threshold inquiries: the level of ordinary skill in the art must be established; the scope and content of the prior art must be determined; and any differences between the prior art and the claims at issue must be ascertained. *Graham v. John Deere Co.*, 383 U.S. 1, 17-18 (1966).

1. Claims Cannot Be Found Obvious if an Element Is Absent.

If a single element of the claim is absent from the prior art, the claim cannot be considered obvious. *See CFMT, Inc. v. YieldUp Int'l Corp.*, 349 F.3d 1333, 1342 (Fed. Cir. 2003) ("Obviousness requires a suggestion of all limitations in a claim.") (citing *In re Royka*, 490 F.2d 981, 985 (C.C.P.A. 1974)); *In re Rijckaert*, 9 F.3d 1531, 1534 (Fed. Cir. 1993) (reversing obviousness rejection where prior art did not teach or suggest all claim limitations); *Garmin Int'l, Inc. v. Patent of Cuozzo Speed Techs. LLC*, IPR2012-00001, Paper 15 at 15 (PTAB Jan. 9, 2013) (refusing to institute an *inter partes* review under 35 U.S.C. § 103 where prior art did not disclose all claim limitations).

Additionally, it is improper to use the challenged claim as a roadmap to piece together disparate disclosure using hindsight bias. *See InTouch Techs., Inc. v. VGo Comm'ns., Inc.*, 751 F.3d 1327, 1351 (Fed. Cir. 2014) (reversing obviousness finding where "[i]t appears that [the expert] relied on the [] patent itself as her roadmap for putting what she referred to as pieces of a 'jigsaw puzzle' together."); *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 421 (2007) ("A factfinder should be aware, of course, of the distortion caused by hindsight bias and must be cautious of arguments reliant upon ex post reasoning.").

2. *A Petition Must Provide Articulated Reasoning with Rational Underpinning to Combine and/or Modify References.*

The conclusion of obviousness based on a combination of references must also be supported with explicit analysis of a reason to combine those references. *KSR Int'l Co.*, 550 U.S. at 418. The Federal Circuit has stated that such reasons must be more than "mere conclusory statements; instead, there must be some articulated reasoning with some rational underpinning to support the legal conclusion of obviousness" as to the particular combination in the claims. *In re Kahn*, 441 F.3d 977, 988 (Fed. Cir. 2006); *see also KSR Int'l Co.*, 550 U.S. at 418 (explaining that the relevant question is whether there was an "apparent reason to combine the known elements in the fashion claimed by the patent at issue"); *Innogenetics, N.V. v. Abbott Labs.*, 512 F.3d 1363, 1374 (Fed. Cir. 2008) (agreeing with the district court's reasoning that "some kind of motivation must be shown from some source, so that the jury can understand why a person of ordinary skill would have thought of either combining two or more references or modifying one to achieve the patented method"); *LG Elecs., Inc. v. Cellular Commc'ns Equip. LLC*, IPR2016-00197, Paper 7 at 7-11 (PTAB April 29, 2016) (faulting a petition's obviousness analysis for lack of sufficient articulated reasons with rational underpinnings for modifying references to achieve particular elements required in the claims).

VI. SUMMARY OF THE PRIOR ART REFERENCES[3]

A. Overview of Primary Reference: Pike

Pike discloses conventional MapReduce, which is exactly what the ’610 patent seeks to improve upon. Indeed, the parent patent application to Pike (which issued as U.S. Pat. No. 7,650,331) is credited with being a patent on “the principle of distributed MapReduce.” Ex. 2005 at 1. Pike’s Figure 1 depicting its “Large-Scale Data Processing Model” demonstrates this:

---

[3] This Preliminary Response is limited to discussing the independent claims of the ’610 patent. Because Petitioner has failed to carry its burden of demonstrating a reasonable likelihood of success with each of the challenged independent claims, Petitioner has also failed to carry its burden of demonstrating a reasonable likelihood of success with each of the challenged dependent claims. Petitioner only asserts Pike and Chowdhuri against the independent claims. The tertiary reference, MacLeod, is only asserted in combination against certain dependent claims; thus, MacLeod is not discussed herein.

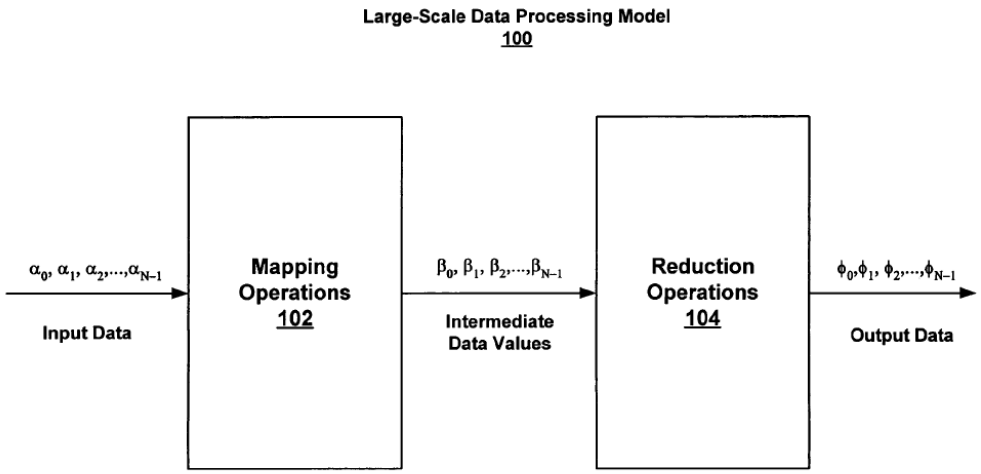


Figure 1

Ex. 1002 at FIG. 1.

Pike explains that "[t]he implementation of large-scale data processing in a parallel and distributed processing environment typically includes the distribution of data and computations among multiple disks and processors to make efficient use of aggregate storage space and computing power." *Id*. at 1:22-26. Pike says that approaches at that time "fail[ed] to provide support for automatically parallelizing these operations across multiple processors in a distributed and parallel processing environment," failed to "automatically handle system faults (e.g., processor failures) and I/O scheduling," and failed to "efficiently handle the analysis of data records." *Id*. at 1:30-35. To address these problems, Pike discloses "[a] method and system for analyzing data records [that] includes allocating groups of records to respective processes of a first plurality of processes executing in parallel. *Id*. at 1:39-41.

The "plurality of processes" in Pike consist generally of "map processes [] to produce a set of intermediate data," and "reduce processes [] to produce output data" from the "intermediate data." *Id*. at 3:54-59. Overseeing these processes is a "work queue master" that performs a number of functions, including "determin[ing] how many map tasks to use, how many reduce tasks to use, which processes and processors to use to perform those tasks, where to store the intermediate data and output data, how to respond to any processing failures, and so on." *Id*. at 4:1-6. Pike's Figure 2 provides a helpful overview that tracks with the '610 patent's discussion of traditional MapReduce. *See* '610 patent at 1:5-27, 1:65-3:34.

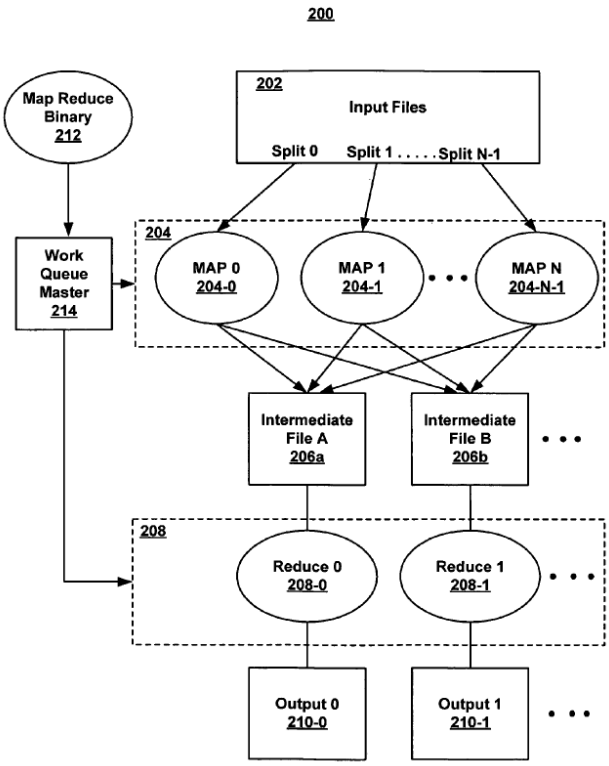


Figure 2

Ex. 1002 at FIG. 2.

In other words, the entire focus of Pike is solving a different problem than that of the ’610 patent—accomplishing conventional MapReduce, which includes the parallelizing of mapping and reducing operations to distribute the processing load. Thus, Pike does not disclose “data groups,” delineating between data based on the “data group” from which the data originated, “intermediate data” that shares a common key with the schemas of the “data groups,” applying MapReduce to data having different schemas to effectuate merger of such data based on a key in common amongst the schemas and the “intermediate data,” or mapping/reducing operations capable of merging data of different schema based on a key in common via processing on a per-“data group” basis. Put differently, Pike is concerned with how to accomplish distributed processing, while the ’610 patent is focused on how to improve upon, and leverage, traditional MapReduce to accomplish merger of heterogenous data. *See* ’610 patent at 3:15-17.

B. Overview of Secondary References: Chowdhuri

Petitioner relies on Chowdhuri as a secondary prior art reference. Chowdhuri discloses a “system and methodology for parallel query optimization using semantic-based partitioning.” Ex. 1003 at Abstract. The specification explains that, at that time, “existing partitioning strategies [had] a number of limitations,” that “[c]urrent solutions only work[ed] for a small subset of SQL operations,” and that “[e]xisting partitioning techniques [did] not provide a generalized solution applicable to all

database operations, including complex SQL operations such as grouping, joins, distinct operations, unions, and the like." *Id*. at ¶ 13. Chowdhuri thus sought to create "a solution that provide[ed] the ability to efficiently partition data so that multiple subtasks or operations [could] be performed simultaneously." *Id*. at ¶ 14.

In Figure 3A, Chowdhuri depicts a "serial operator tree 300 illustrating a serial plan for executing a query," where "order and customer tables (data 308, 309) are scanned by two separate scan iterators. . . . Once these tables are scanned, results are sent to the hash join iterator," then to a "grouping iterator," and finally to a "sort iterator." *Id*. at ¶ 131.

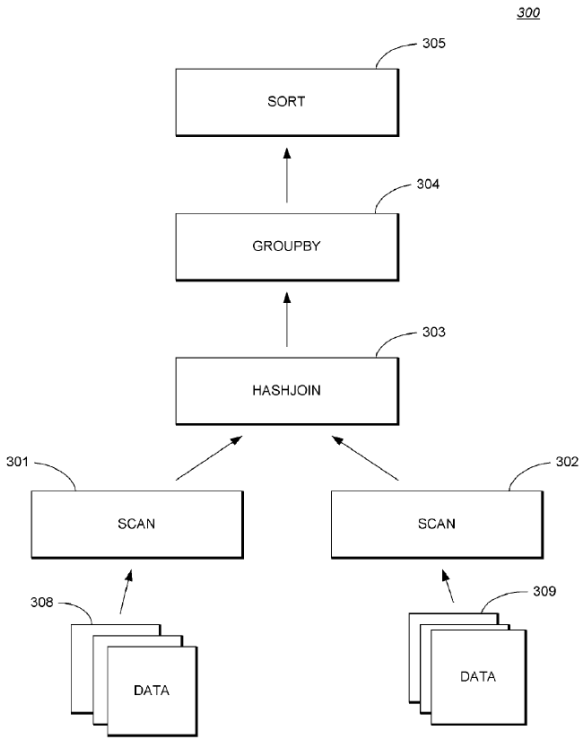


FIG. 3A

Id. at FIG. 3A.

Per Chowdhuri, "[a]n important point about parallelizing a query is to initiate as many iterators as possible and spread them across multiple disk and CPU resources for processing," and Chowdhuri accomplishes this via an "exchange operator." *Id*. at ¶ 133. Chowdhuri's "exchange operator" "permits concurrent execution of complex query evaluation plans in multiple cooperating processes, "allows pipelining of intermediate results from one set of operators (i.e., producer tasks) and initiates another set of operators to process the next request (i.e., consumer tasks)," and "takes the data from the producer tasks and is capable of redistributing data for a different degree of parallelism in its consuming tasks." *Id*. at ¶ 134.

Thus, much like Pike, Chowdhuri is focused on accomplishing parallel processing, which is not the problem the '610 patent seeks to solve. And as discussed further below, Chowdhuri discloses an entirely different mechanism than MapReduce to process data, namely, using the exchange iterators discussed above to distribute portions of original input data to hash join functions that consolidate data. Chowdhuri does not disclose "data groups," delineating between data based on the "data group" from which the data originated, providing each partition of each "data group" to a "selected one" mapping function, "intermediate data" that shares a common key with the schemas of the "data groups," applying MapReduce to data having different schemas to effectuate merger of such data based on a key in common amongst the schemas and the "intermediate data," or mapping/reducing

operations capable of merging data of different schema based on a key in common via processing on a per-"data group" basis.

VII. THE PETITION SHOULD BE DENIED UNDER 35 U.S.C. § 325(d).

The Director has discretion to deny institution of an *inter partes* review proceeding under 35 U.S.C. § 325(d) when "the same or substantially the same prior art or arguments previously were presented to the Office." *Advanced Bionics*, Paper 6 at 7. The Board applies a two-part framework under § 325(d): "(1) whether the same or substantially the same art previously was presented to the Office or whether the same or substantially the same arguments previously were presented to the Office; and (2) if either condition of the first part of the framework is satisfied, ***whether the petitioner has demonstrated*** that the Office erred in a manner material to the patentability of challenged claims."[4] *Advanced Bionics,* Paper 6 at 8 (emphasis added).

---

[4] Art previously presented includes "art made of record by the Examiner, and art provided to the Office by an applicant, such as on an Information Disclosure Statement (IDS), in the prosecution history of the challenged patent." *Advanced Bionics*, Paper 6 at 8. If the first prong of factor (1) is met (i.e. substantially the same

A. Substantially the Same Art and Arguments Were Previously Presented to the Office.

The prior art asserted by Petitioner is substantially the same as art previously presented to the Office. Just like in *Advanced Bionics*, Petitioner here relies on references that, while not themselves before the Office during prosecution, nonetheless disclose substantially the same information as references cited and analyzed during prosecution.

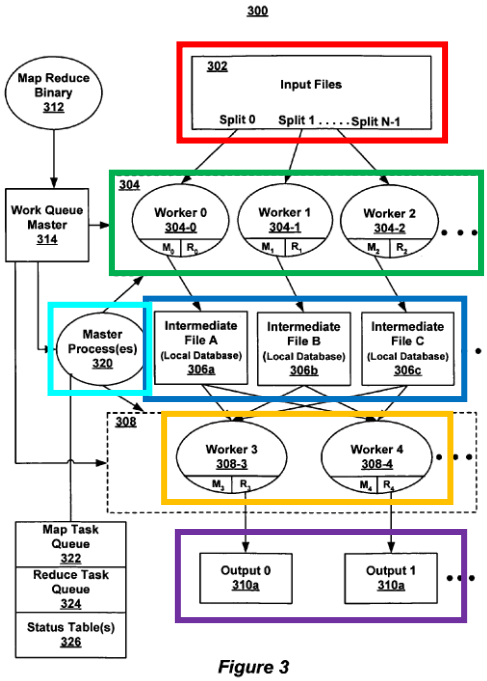
The Petition relies on Pike for both grounds asserted against the independent claims, arguing that Pike renders the claims obvious by itself and/or in combination with Chowdhuri. *See* Petition at 2. Petitioner describes Pike as disclosing "worker processes (304, 308) in large-scale data processing system 300 that invoke mapping and reducing operations." *Id*. at 10 (citing Ex. 1002 at 8:65-11:55, Fig. 3). Petitioner also points to Pike's Figure 3 and explains that Pike discloses that, over a distributed system, mapping operations are applied to input data to generate intermediate data, and that reduce operations reduce the intermediate data to provide output data. *Id*. at 10-11. In short, Petitioner explains that Pike discloses conventional MapReduce.

---

art was previously presented), then no analysis is needed regarding whether the same arguments were previously presented to the Office. *Id.* at 20.

The relevant disclosure in Pike is substantially similar, if not identical, to art that was before the USPTO during prosecution. To explain conventional MapReduce and demonstrate its shortcomings, the ’610 patent thoroughly discusses a 2004 paper by Jeffrey Dean and Sanjay Ghemawat (Ex. 1009) and a related presentation (Ex. 1011) that each cover traditional MapReduce. *See* ’610 patent at 1:6-27; 1:66-3:17. The ’610 patent then proceeds to explain how the disclosed inventions are different and improve upon traditional MapReduce architecture. *See, e.g., id*. at 3:15-64. In addition to being discussed in the specification, the paper was included in an IDS and listed as a reference cited in the patent. *See* Ex. 2006 at 2. In fact, Figures 1 and 2 of the ’610 patent were copied from the Dean/Ghemawat presentation. *Compare* Ex. 1001 at FIGS. 1 and 2, *with* Ex. 1011 at 7-8. There can thus be no doubt that the Dean/Ghemawat paper and presentation were front-and-center during prosecution.

Furthermore, it cannot be disputed that Pike corresponds to the Dean/Ghemawat references—they each disclose the same thing, namely, conventional MapReduce. This is unsurprising considering that both Dean and Ghemawat are named co-inventors with Robert Pike on the Pike reference. The similarities are unmistakable. For instance, Figure 3 in Pike is strikingly similar to Figure 1 in the Dean/Ghemawat paper, which is emphasized by the colored annotations below:



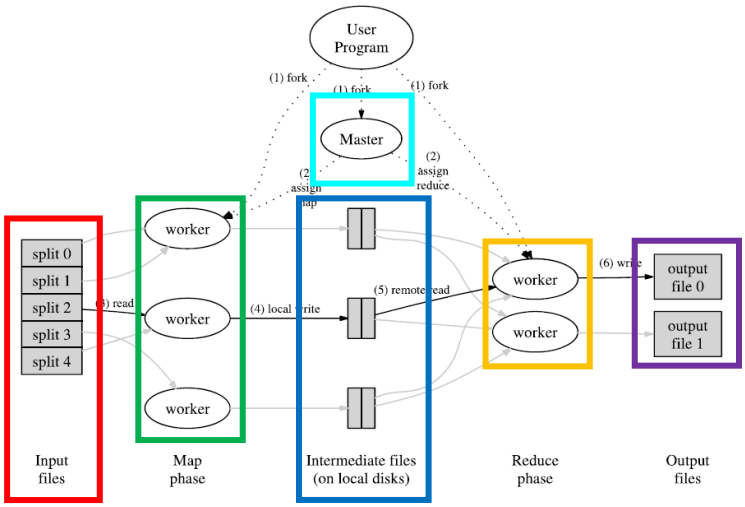


Figure 1: Execution overview

Ex. 1002 at FIG. 3 (annotated) Ex. 1009 at 3 (annotated)

Moreover, Dean/Ghemawat explain that the user provides "the names of the input and output files" (Ex. 1009 at 2); Pike discloses the same thing (*see* Petition at 17-18, 37). Dean/Ghemawat discuss "partitioning the input data into a set of *M splits*" to be "processed in parallel by different machines" (Ex. 1009 at 3-4); Pike discloses the same thing (*see* Petition at 21-22). Dean/Ghemawat discuss a "master" that stores in a data structure the "state (*idle*, *in-progress*, or *completed*)" for "each map and reduce task" and "the locations and sizes" of the intermediate files, and further that input/output files are tracked by the master for the particular tasks (Ex. 1009 at 4); Pike discloses the same thing (*see* Petition at 28-33). In essence, *every*

feature that Petitioner indicates in Pike purportedly rendering the claimed inventions obvious is also disclosed in Dean/Ghemawat.

Chowdhuri is also substantially similar to the art considered by the USPTO, and Petitioner's arguments mirror those of the examiner (which the applicant overcame). For example, Petitioner points to Chowdhuri as disclosing tables with different schema (*see* Petition at 38-42); U.S. Pat. No. 6,341,289 to Burroughs (Ex. 2007) was cited by the USPTO during prosecution as teaching the same feature. *See* Ex. 1004 at 155. And, unlike Chowdhuri, Burroughs actually uses the term "schema." *See generally* Ex. 2007. Further, the examiner made the exact same argument that Petitioner is making here, namely, that combining the different schemas in Burroughs with a MapReduce implementation renders the claims obvious. *See* Ex. 1004 at 150. Such is precisely what Petitioner alleges here, only Petitioner substituted Chowdhuri for Burroughs. *See* Petition at 38-42. As another example, U.S. App. Pub. No. 2006/0117036 to Cruanes (Ex. 2008) was cited by the USPTO as allegedly teaching data groups with different schemas. *See* Ex. 1004 at 113-114. And just like Chowdhuri, Cruanes is focused on ***scanning in portions of tables and performing hash joins of scanned-in data in parallel.*** *See* Ex. 2008 at ¶¶ 6-18. This is *exactly* what Petitioner relies upon Chowdhuri to disclose; Chowdhuri discloses nothing on this point that was not already considered by the USPTO.

As another example, Petitioner relies on Chowdhuri for teaching processing "intermediate data" "in a manner that is defined to correspond" to its parent "data group" because Chowdhuri discloses "joining, grouping, and sorting based on customer and order 'state.'" Petition at 49. The examiner made an identical argument using Cruanes, alleging Cruanes teaches joining tables based on a matching "deptID" value. *See* Ex. 1004 at 76-77, 81, 83-85; *see also* Ex. 2008 at ¶ 5. Applicant overcame this argument in prosecution by demonstrating that Cruanes "merely provide[s] a joining function and [does] not provide a plurality of mapping functions that output lists for keys found in the data partition, in the manner claimed." Ex. 1004 at 69. This is precisely the issue with Chowdhuri's disclosure discussed below, i.e., that Chowdhuri simply discloses hash join functions, not mapping functions that output "intermediate data." Applicant's arguments were successful, and the claims were allowed. In summary, every feature of Chowdhuri to which Petitioner points was disclosed in the prior art analyzed by the USPTO, and Chowdhuri is, thus, substantially similar to art before the USPTO during prosecution.

B. Petitioner Has Failed to Demonstrate that the Office Committed Material Error.

Because substantially the same art was already presented to the Office, the Board should determine "whether the petitioner has demonstrated that the Office erred in a manner material to the patentability of challenged claims." *Advanced Bionics*, Paper 6 at 8 (emphasis added). In *Advanced Bionics*, the petitioner made

arguments that the examiner erred during prosecution in evaluating prior art, referring to specific technical aspects and claim limitations that the petitioner asserted the examiner had failed to properly analyze and research. *See id*. at 20. Even though the petitioner there specifically pointed out how it thought the examiner erred and cited additional references that potentially disclosed relevant technical aspects, the Board nonetheless concluded that the petitioner was not persuasive in showing that "the Examiner materially erred in determining the patentability of the challenged claims." *Id*. at 22.

Here, Petitioner fails to demonstrate that "the Office erred in a manner material to patentability of the challenged claims." *Id*. at 8. In fact, Petitioner does not assert that the Office erred at all. Unlike in *Advanced Bionics*, where the Board found that petitioner failed to persuasively demonstrate error when the petitioner actually presented arguments, Petitioner here has not attempted to demonstrate error. *See id*.

Petitioner's failure to address the *Advanced Bionics* framework here is particularly egregious. American Airlines, Inc. and Hilton Domestic Operating Company, Inc. filed a petition against the '610 patent last year. It is obvious that Petitioner reviewed that petition and R2's preliminary response. Indeed, the current Petition sets forth identical unpatentability grounds against the independent claims, responds to arguments R2 included in its previous preliminary response, almost

whole-sale copies the previous petition's table of exhibits, and adopts the previous petition's substantive arguments. *Compare American Airlines, Inc. et al. v. R2 Solutions LLC*, IPR2023-00689, Paper 2 (PTAB Mar. 7, 2023), *with* Petition. *Advanced Bionics* was a central issue raised in R2's response to AA's and Hilton's petition—the Board requested additional briefing on precisely that issue. *See American Airlines*, IPR2023-00689 at Papers 13-14, 17. And yet despite having actual knowledge of the *Advanced Bionics* issue, Cloudera declined to argue the Office erred or otherwise mention *Advanced Bionics* at all. In fact, it appears that Cloudera is counting on being given the opportunity to address this issue in additional briefing, as opposed to addressing the issue in the Petition as it should have. This is fatal to Cloudera's Petition. Because substantially the same information was previously before the Office, and because Petitioner has failed to demonstrate that the Office committed material error, the Board should deny institution under § 325(d).

VIII. THE PETITION DOES NOT DEMONSTRATE THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE.

To justify the institution of an *inter partes* review, a petitioner must establish that there is a "reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition." 35 U.S.C. § 314(a). Petitioner has not met its burden of establishing a reasonable likelihood of success at least because the Petition fails to show that each limitation of the independent claims is

disclosed or taught by the asserted prior art references of each ground. Consequently, the Board should deny the Petition and decline to institute the *inter partes* review.

As discussed above, the ’610 patent claims a stem-to-stern augmentation to MapReduce, requiring specialized data structures (“data groups”) at the front-end configured to be merged together via selective dissemination amongst and processing by a collection of mapping and reducing functions within the enhanced MapReduce architecture. This in turn requires the mapping functions to be specialized so that they can be “selected” for a partition based on the “data group” the partition originated from and map the partitions “differently” based on “data group” to form “corresponding intermediate data.” It also requires a reducing phase configured to distinguish the “intermediate data” based on the parent “data groups” to effectuate specialized processing on a per-“data group” basis. These enhancements are reflected as requirements of each independent claim. *See* Ex. 1026. Claims 17 and 40 mirror the requirements of Claims 1 and 33, respectively. *See id.*

These features are not disclosed by Pike or Chowdhuri. Petitioner’s first argument is that Pike discloses conventional MapReduce and thus renders the independent claims obvious (Ground 1). This is wrong, as the ’610 claims an *improvement* to traditional MapReduce. Petitioner’s second argument is that Chowdhuri discloses a way to join tables, and combining Chowdhuri with Pike

teaches all of the limitations of the independent claims (Grounds 2 and 3). But Petitioner ignores the requirements of the claims and glosses over that there is no workable combination of Pike and Chowdhuri that could arrive at the challenged claims—the references teach different and conflicting mechanisms. Tellingly, as discussed above, the USPTO found the challenged claims to be patentable over a near-identical combination of references during prosecution. The Petition should be denied.

A. Ground 1: Pike Does Not Teach or Suggest at Least One Limitation of Each Independent Claim.

1. Pike Does not Disclose "Data Groups."[5]

Each of Claims 1, 17, 33, and 40 requires "data groups," and Petitioner is wrong that Pike discloses or renders obvious "data groups." *See* Petition at 16-17.

---

[5] The limitations relevant to this argument are: [1b], [17a], and [40a] ("wherein the data set comprises a plurality of data groups"); [33a] ("A map-reduce method of processing data from a plurality of groups having different schema over a computer system"); [33b] and [40b] ("for a first data set having a plurality of first key-value pairs, wherein such first data set belongs to a first data group and the first key-value pairs have a first schema"); and [33f] and [40f] ("for a second data set having a

Relying solely on its expert's declaration, Petitioner alleges that the input files of Pike are "data groups" because they are identifiable by "file name." *Id*. at 17. But conflating input files with "data groups" ignores the Eastern District of Texas's construction of "data groups" that Petitioner purportedly applies. Nowhere does Pike teach "a mechanism for identifying *data*" from any particular input file. The file name certainly does not meet this requirement, because the file name *only* identifies the file itself, which is nothing more than the *location* of the data to be processed.

Indeed, Pike discloses that a function of the "work queue master" is to determine "where to store" data, i.e., in which files. Ex. 1002 at 4:1-6. The file name in Pike is not disclosed as identifying the *data* that comes from that file such that the programming architecture can accomplish specific processing based on the file name. The only disclosed function of the file name in Pike is to allow a user to specify which files to process, or, in other words, the location of the data to be processed. *See id*. at 4:34-40 ("[T]o perform a specific data processing operation on a set of input files, the only information that must be provided by an application programmer is: information identifying the input file(s), information identifying or

---

plurality of second key-value pairs, wherein such second data set belongs to a second data group and the second key-value pairs have a second schema").

specifying the output files to receive output data, and two application-specific data processing operators, hereinafter referred to as map( ) and reduce( ).") And all that Pike does with a given file name beyond using it to locate which data to process is include it in a "task status table" for the work queue master to "keep[] track of the status of map and reduce tasks." *Id*. at 55-56. Nowhere does Pike disclose identifying *data* from an input file with the file name in the course of processing such data, meaning that Pike's file names cannot be considered "a mechanism for identifying *data* from that [data] group." [6]

Pike's usage of file names is consistent with the problem that Pike aims to solve and emphasizes why Pike does not disclose "data groups" or render them obvious. The entire purpose of Pike is to accomplish the distribution and

---

[6] Petitioner's claim that an input file name would include "an extension indicating its file type or structure" is inapposite for two reasons. First, Pike never discusses file extensions at all, so Petitioner's discussion on this point is purely conjecture. Second, even if Pike's file names had extensions, the extensions only identify data type, not the actual data. Regardless, Pike only discloses using the file names to identify the location of the data to be processed, not to identify the data itself throughout the mapping and reducing phases.

parallelization of MapReduce processing. To do this, Pike's architecture implements the "work queue master" to "assign[] tasks to processes," and Pike discloses that "multiple processes may be executed by each of the processors in the group of processors that are available to do the work assigned by the work queue master." *Id*. at 4:7-11. Thus, the "work queue master" must be capable of effectuating distribution of the processing load, which includes directing input/output to particular nodes and processes and tracking the status of each available process. Pike discloses receiving file names from users to determine which files need to be processed and subsequently distributing processing of the particular files amongst its architecture. But what Pike does *not* disclose is how to use "data group" identifiers to deal with data of different schema within the actual data processing steps to accomplish merger of the data of such "data groups." Indeed, Pike is unconcerned with the *type* of data being processed because Pike is not trying to solve the problem of applying its architecture to data of different schema to merge heterogenous data.

Petitioner is also wrong that a "POSITA would have found it obvious" to use input files as data groups "because it would have allowed for locating that data group in Pike's system[,] understanding the data in that file … and write 'application-specific' map and reduce functions on those files." Petition at 18. Petitioner is using the claims as a roadmap. Pike's discussion of "application-specific data processing

operations" is limited to disclosing that an operator could "be required to process the input data one record at a time," or could "be required to generate its output data in the form of key/value pairs." Ex. 1002 at 7:31-36. Pike simply does not disclose or even suggest an "application-specific" operator designed to use an input file name to identify data from that file and adjust processing accordingly, so Pike cannot render this limitation obvious.

2. *Pike Does not Disclose Delineating "Data Groups" Prior to Partitioning.*[7]

The Petition fails to demonstrate that Pike teaches delineating data groups prior to partitioning. Claims 1 and 17 require that the "partitioning" occur on "the

---

[7] The limitations relevant to this argument are: [1c] and [17b] ("partitioning the data of each one of the data groups into a plurality of data partitions that each have a plurality of key-value pairs"); [33a] ("A map-reduce method of processing data from a plurality of groups having different schema over a computer system"); [33b] and [40b] ("for a first data set having a plurality of first key-value pairs, wherein such first data set belongs to a first data group and the first key-value pairs have a first schema"); [33c] and [40c] ("partitioning the first data set into a plurality of data partitions"); and [33f] and [40f] ("for a second data set having a plurality of second

data of each one of the data groups," and Claims 33 and 40 require that the first and second data sets belong to first and second "data groups," respectively. Thus, in all claims, the "data groups" are delineated prior to partitioning. Pike does not disclose or suggest this. While partitioning data was certainly a step in conventional MapReduce and is contemplated by Pike, Petitioner's (incorrect) argument is that Pike's "input files" are "data groups." So for Pike to disclose this limitation, each "input file" would have to partitioned into a "plurality of data partitions." But this is not what Pike teaches. Instead, Pike just discusses splitting a collection of "input files" based on a "specified or predefined size" of the desired "data blocks" (Ex. 1002 at 4:55-56), and Pike teaches that a "data block" can include data from multiple "input files" and that an "input file" can be its own "data block." *See id*. at 6:59-72 ("In some embodiments, the input files field may specify a portion of an input file (e.g., where the portion comprises a data block) to be processed by the task, or this field may specify portions of two of more input files."); 4:58-59 ("Alternately, in some embodiments . . . the individual files are the data blocks."). In other words,

---

key-value pairs, wherein such second data set belongs to a second data group and the second key-value pairs have a second schema").

Pike teaches both that the "input files" do not have to be partitioned, and that the "data blocks" can be formed from data of multiple "input files."

This teaching in Pike is contrary to the claims, which require "***partitioning the data of each one of the data groups*** into a plurality of data partitions" (for Claims 1 and 17) and that the first and second data sets ***belong to first and second "data groups,"*** respectively, prior to partitioning (for Claims 33 and 40). The "data blocks" in Pike are not formed for "each one of the" of the "input files" and do not "belong" to any sort of data group, as is required. Instead, Pike's "data blocks" can be entire "input files," parts of "input files," or parts of multiple "input files," and are thus not "partitions" formed for "each one of" the "input files" and do not "belong" to any delineated grouping of "input files." Indeed, Pike splits an amalgamation of "input files" into "data blocks" without any grouping of the files, by file name or otherwise, prior to the split. This is reflected in Pike's figures 2 and 3, which depict the "input files" as a collection of data that are not grouped in any manner prior to the split.

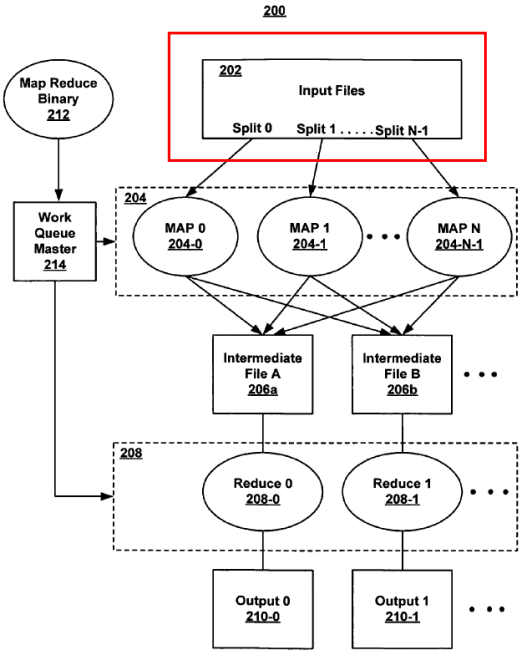


Figure 2

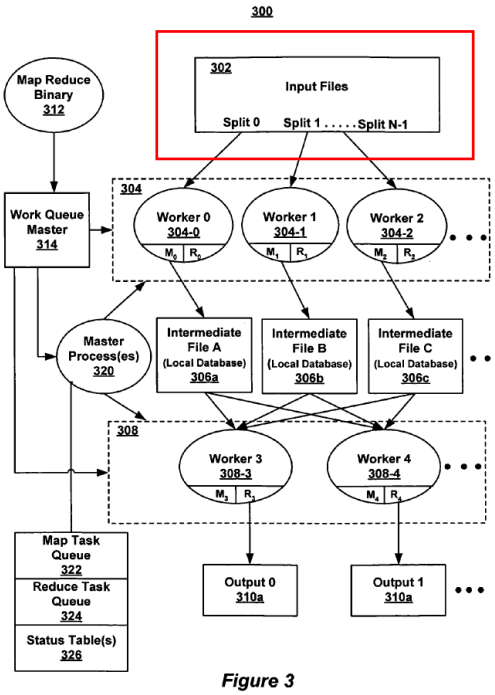


Figure 3

Id. at FIG. 2, FIG. 3 (annotated).

In short, Pike never contemplates any delineation of groups of "input files" or delineation of individual "input files" prior to splitting the files into "data blocks," emphasizing that (1) the "input files" in Pike cannot be "data groups" within the meaning of the '610 patent, and (2) Pike does not teach "partitioning the data of each one of the data groups into a plurality of data partitions" (for Claims 1 and 17) or that the first and second data sets belong to first and second "data groups," respectively, prior to partitioning (for Claims 33 and 40). *Id*. at 4:54-55.

3. Pike Does not Disclose "Intermediate Data" that Is "Identifiable" to a "Data Group."[8]

Pike fails to disclose or render obvious "intermediate data" that is "identifiable" to the "data group" from which the "intermediate data" came, as is required by each of Claims 1 and 17. Petitioner points to the "task status table" in Pike as purportedly disclosing this limitation, alleging that "Pike[] track[s] input files through the map and reduce processes" and "generat[es] intermediate data that is identifiable to a particular input file…." Petition at 31-32. But Petitioner both mischaracterizes Pike and ignores the requirements of the claim language. While Pike certainly tracks which files are allocated to and generated by particular processes, there is no indication in Pike that the "intermediate *data*" itself is "identifiable" to a given "input file." To the contrary, Pike's "task status table" simply reveals that the "work queue master" is aware of (1) which "input files" were handled by which processes, and (2) in which "output files" the resulting "intermediate data" is stored.

---

[8] The limitations relevant to this argument are: [1e] and [17d] ("to independently output a plurality of lists of values for each of a set of keys found in such map function's corresponding data partition to form corresponding intermediate data for that data group and identifiable to that data group").

Task Status Table

Task ID	Status	Process	Input Files	Output Files	...
Map0000	Completed	P0000	12340	12341	...
Map0001	Completed	P0001	12344	12345	...
⋮	⋮				
Map0103	In Progress	P0010	12030	12031	...
Map0104	Failed	P0011	10101	12102	...
Map0103b	In Progress	P0010	12030	12031	...
⋮	⋮				
Red0000	In Progress	P0033	12340, ...	14000	...
Red0001	Waiting	-	-	-	...

Figure 7A

Ex. 1002 at FIG. 7A.

In other words, while Pike tracks which "output file" is generated from processing given "input files," Pike is silent with respect to whether the alleged "intermediate *data*" itself that is stored in the "output file" is identifiable to any original "input file." In fact, ensuring that "intermediate data" is "identifiable" to a given "input file" is contrary to Pike's teachings. Pike discloses that a particular process "retrieves data from ***one or more input files*** (or the one or more input file portions) identified in the input file field and writes the results of the task ID to ***one or more output files*** identified in the output file field." *Id*. at 6:65-7:2. Thus, a given "output file" can contain "intermediate data" stemming from multiple "input files." But far from teaching that the "intermediate data" in an "output file" (which originates from at least two different "input files" in this example) is "identifiable" to their particular "input files," Pike teaches that all of the "intermediate data" is amalgamated into an "output file" with a file name that is *different* from that of the

"input files." *See id*. at FIG. 7A (depicting the "output file" name as different from the "input file" name). Indeed, per Pike, the "intermediate data" *loses* its identification with its respective "input files," because all "intermediate data" generated by a given process, regardless of where it originated from, is simply entered into a single "output file" with a new file name. *See id*.

Petitioner attempts to remedy the shortcomings in Pike by arguing that "it would have been obvious to include an identifier associated with Pike's input files to track them through the map and reduce tasks." Petition at 32. But this is an about-face—indeed, Petitioner previously alleged that Pike already discloses such an identifier, i.e., Pike's "file name," which is the purported "mechanism for identifying data." *See id*. at 16-18. Petitioner does not allege here that the "file name" is the "identifier" to track data "through the map and reduce tasks" and instead argues that it would just be obvious to include one. That is because the "file name" *cannot* serve as this mechanism for two reasons: (1) Pike's "data blocks" (purported "partitions") can be entire "input files," parts of "input files," or parts of multiple "input files," meaning that a file name from just one of these input files cannot identify all of the data within a data block; and (2) Pike's "intermediate data" *loses* its identification with its respective "input files" as discussed above, meaning that the original "file names" of the "input files" are of no moment to the actual data processing taking place in Pike.

In any event, Petitioner's argument that it would be obvious to modify Pike in this fashion is fraught with hindsight. Petitioner claims that modifying Pike this way "would improve efficiency of the system by enabling a data set to be selectively reduced after it is mapped, without requiring the desired subset of input files to be re-mapped." *Id*. at 33. But Petitioner is just reciting a benefit provided by the '610 patent and claiming that it would be obvious to modify Pike to achieve it. Petitioner cannot show that there is any teaching in Pike or elsewhere that suggests that Pike could or should be modified like this. Contrary to Petitioner's argument, Pike's status tables certainly do not suggest such a modification, because, as discussed above, the status tables only reveal that Pike tracks which "output file" is generated from processing given "input files," not that any data itself in that file is identifiable to any original "input file." Nothing in Pike teaches that "intermediate data" should be identifiable to an original input file in data processing. And this makes sense, because Pike is only focused on providing a distributed architecture for generic MapReduce and not on joining data of different schema via mapping and reducing. This emphasizes why Pike does not disclose "data groups" within the meaning of the '610 patent and also fails to disclose or render obvious this claimed feature.

4. Pike Does not Disclose "Data Groups" with Different Schemas.[9]

Pike does not disclose or render obvious "wherein the data of a first data group has a different schema than the data of a second data group" (as required by Claims 1 and 17) or "wherein the first schema differs from the second schema" (as required by Claims 33 and 40). Indeed, Pike does not even use the term "schema."

Petitioner first argues that Pike's disclosure that its "input files . . . can include a variety of data types" discloses data groups with different schemas. *See* Petition at

---

[9] The limitations relevant to this argument are: [1f] and [17e] ("wherein the data of a first data group has a different schema than the data of a second data group"); [33a] ("A map-reduce method of processing data from a plurality of groups having different schema over a computer system"); [33b] and [40b] ("for a first data set having a plurality of first key-value pairs, wherein such first data set belongs to a first data group and the first key-value pairs have a first schema"); [33f] and [40f] ("for a second data set having a plurality of second key-value pairs, wherein such second data set belongs to a second data group and the second key-value pairs have a second schema"); [33i] and [40i] ("wherein the first schema differs from the second schema"); and [33l] and [40k] ("the output data set has a different schema than the first and second schema").

36. But disclosing that "input files" can be of different types falls short of disclosing this element. Claims 1 and 17 require an input "data set" with a "plurality of data groups," and that a first and second "data group" of this "data set" have different schemas; Claims 33 and 40 similarly require first and second "data groups" with schemas that are different. Pike simply does not teach this. Indeed, there is *no* requirement in Pike that the "input files" have different schemas, and Pike's disclosure that the "input files" can be any variety of data types just emphasizes the schema of the data processed by Pike's architecture is irrelevant. Indeed, as recognized by Petitioner, "MapReduce programmers" utilizing Pike's system "need to know the input files, where to send output data from the map and reduce processes," and "two application-specific data processing operators, hereinafter referred to as map() and reduce()." Petition at 37 (internal quotations omitted). Notably absent from this list is the schema of the data being input. Again, this is because Pike is not focused on how to join data of different schema, so the schema is of no moment to Pike.

Petitioner then argues that Pike's discussion of "application-specific" map and reduce functions renders this feature obvious. Not so. As discussed above, Pike's discussion of "application-specific data processing operations" is limited to disclosing that an operator could "be required to process the input data one record at a time," or could "be required to generate its output data in the form of key/value

pairs." Ex. 1002 at 7:31-36. Nowhere does Pike suggest an "application-specific" operator designed to handle "data groups" with different schema by using file names as identifiers for data coming from those files. In fact, Pike teaches away from such functionality because, as discussed above, Pike teaches that "intermediate data," which come from one or more input files, loses its identification with the original input files—this means that "application-specific" reduce operators would have no ability to delineate between data of different schema in the reduce phase based on the original input file names.

Additionally, Petitioner repeatedly alleges that it would have been obvious to apply Pike's mapping and reducing operations to diverse input files and/or to input data with different schema. *See id*. at 36-38. But this ignores the claimed features of the '610 patent. *Id*. at 40. The '610 patent does not claim simply "[a]nalyzing input files of a different schema" (*id*. at 37)—instead, the claims offer a *specific mechanism* by which MapReduce can be efficiently applied to merge data having different schemas. *Id*. Claims 1 and 17 require that there is an input "data set" comprising a "plurality of data groups" (at least two of which have different schemas) that are each partitioned and processed to form "intermediate data" that is "identifiable" to their particular "data groups." Similarly, Claims 33 and 40 require delineating input data sets into first and second "data groups" having key-value pairs with schemas that are different. From a high level, Pike plainly does not do this, and

it also does not provide any sort of teaching that would lead a POSITA to modify Pike in this fashion. This is apparent from the lack of any discussion in Pike regarding the schema of the input data or any grouping of the input data prior to mapping. It is also apparent from Pike's lack of disclosure regarding forming "intermediate data" that is "identifiable" to any particular input file to affect reduction that corresponds to any particular input file.

Additionally, with respect to Claims 33 and 40, Pike does not disclose that an "output data set" from a reduce function "has a different schema than the first and second schema" of the input sets of key-value pairs, because Pike does not discuss "schema" at all, disclose what the schemas of the input sets are, or disclose what the schemas of the output sets are.

5. Pike Does not Disclose a Key in Common Between the (1) "Data Groups"/Schema of the Input Data, and (2) the Intermediate Data.[10]

Petitioner fails to establish that Pike discloses or renders obvious a key in common between (1) "data groups" (for Claims 1 and 17) or schema of the input

---

[10] The limitations relevant to this argument are: [1h] and [17g] ("wherein the different schema and corresponding different intermediate data have a key in

data (for Claims 33 and 40), and (2) the intermediate data. Petitioner first alleges that Pike's disclosure of a reduce function that "merges or otherwise combines the sorted intermediate data values having the same key" meets these requirements. Petition at 46. But in addition to ignoring the requirement that the "data groups" or input data sets have "different schema," this argument also ignores that "the different schema" of the "data groups"/input data sets, as well as the corresponding intermediate data, must all have "a key in common." Pike does not disclose this. Instead, Pike just discloses that the collections of "intermediate data" all have the same key. Nowhere does Pike disclose that the "input files" (which Petitioner alleges are "data groups") have a key in common with each other, or with the "intermediate data" that is generated by processing the "input files." Pike thus does not disclose this limitation or render it obvious.

Petitioner is also wrong that "[t]o reduce on 'the same key,' the input data and the intermediate data must have "a key in common." Petition at 46. Indeed, in traditional MapReduce, the entire purpose of the mapping functions was to *generate* common keys such that the reduce function could reduce the data based on that key.

---

common"); and [33j] and [40l] ("wherein the first and second schema and the first and second set of resulting key-value pairs have a key in common").

This is demonstrated by Pike's discussion of using a "map-reduce operation for counting the number of occurrence of each distinct word in a large collection of documents." Ex. 1002 at 10:37-39. Pike discloses that the key/value pair generated by the map functions would be "<w, 1>," with the "w" corresponding to an identified word and acting as the key. *Id*. at 10:39-53. The reduce function then "adds up the count values" based on the generated, common keys, i.e., the identified words. *Id*. In other words, Pike discloses generating intermediate data with common keys created via mapping functions, and then reducing the data based on the generated common keys. But nowhere does Pike disclose that "input files" share a common key with one another or with the "intermediate data" prior or subsequent to the mapping phase.

6. Pike Does not Disclose Processing Intermediate Data According to its "Data Group," or Merging the Intermediate Data Based on a Key in Common Between the Intermediate Data and the "Data Groups."[11]

Pike fails to teach "processing the intermediate data for each data group in a manner that is defined to correspond to that data group . . . so as to result in a merging

---

[11] The limitations relevant to this argument are: [1i] and [17h] ("[reducing/reduce] the intermediate data for the data groups to at least one output data group, including

of the corresponding different intermediate data based on the key in common," as is required by Claims 1 and 17. As discussed above, Pike does not disclose "data groups" that have a "key in common" with each other and the "intermediate data," so Pike cannot teach "merging" based on such a common key. Further, Pike does not disclose that processing "intermediate data" in the reduce phase occurs "in a manner that is defined to correspond" to the "data groups" from which the "intermediate data" originated.

Again, Petitioner alleges that "input files" in Pike are "data groups." So for Pike to disclose "processing the intermediate data for each data group in a manner that is defined to correspond to that data group," Pike would have to show that in the reduce phase, the "intermediate data" is processed "in a manner that is defined to correspond" to the "input file" from which the "intermediate data" came from. But this is not what Pike teaches. Instead, Pike just teaches tracking each reduce operation associated with an input file. Ex. 1002 at 11:33-38, 13:5-17, 6:55-7:12. But tracking which operations handle which files is not the same as *processing* the

---

processing the intermediate data for each data group in a manner that is defined to correspond to that data group, so as to result in a merging of the corresponding different intermediate data based on the key in common").

data "*in a manner that is defined to correspond*" to the original input file. Indeed, nowhere does Pike disclose that the original input file defines how "intermediate data" from that file is processed, and Pike thus does not disclose the claimed feature. And Pike's disclosure of "application-specific" processes does not remedy this deficiency, because again, Pike's discussion of "application-specific data processing operations" is limited to disclosing that an operator could "be required to process the input data one record at a time," or could "be required to generate its output data in the form of key/value pairs." *Id*. at 7:31-36. Pike does not disclose "application-specific" operators designed to use an input file name to identify data from that file and adjust processing accordingly. Pike thus does not disclose this limitation or render it obvious.

B. Grounds 2 and 3: Combining Chowdhuri with Pike Fails to Render the Independent Claims Obvious.[12]

1. Chowdhuri Does not Disclose "Data Groups."[13]

Chowdhuri does not teach "data groups," and thus the combination of Pike with Chowdhuri fails to render these limitations obvious. Just as Pike's input files are not "data groups," Chowdhuri's "order" and "customer" tables are also not "data groups." Petitioner is wrong that Chowdhuri's table names transform Chowdhuri's data tables into "data groups." Petition at 18. Just as in Pike, Chowdhuri only discloses using table names to determine which data to process, not as mechanisms for identifying data from those tables in the course of processing such data. Indeed, there is no disclosure in Chowdhuri of any sort of identifier (table name or otherwise) being, e.g., passed to specific functions to guide group-specific processing as in the

---

[12] In Ground 2, Petitioner contends that all claims are rendered obvious by Pike in view of Chowdhuri. In Ground 3, Petitioner contends that certain dependent claims are rendered obvious by Pike and Chowdhuri in view of MacLeod. Because Pike in view of Chowdhuri does not render any of the independent claims obvious, Pike and Chowdhuri in view of MacLeod do not render any of the dependent claims obvious. This discussion accordingly applies to both Grounds 2 and 3.

[13] The same limitations relevant to Section VIII.A.1 are relevant here.

’610 patent. *See*, *e.g.*, Ex. 1001 at 6:29-33, 39-48; 7:45, 55, 64-65; FIG 5. To the contrary, Chowdhuri’s discussion of vertical and horizontal parallelism (which Petitioner relies on) says that the query containing “from customer c, order o” simply indicates that “two tables, the customer table and the order table, are being scanned.” *See*, *e.g.*, Ex. 1003, ¶¶ 126-127. Nowhere in this discussion does Chowdhuri disclose that the processing operations delineate between data from each table via “mechanisms for identifying data.” Thus, Chowdhuri’s tables are not “data groups.” *See id*.

Because neither Pike nor Chowdhuri disclose “data groups,” combining these two references fails to render obvious these limitations. Regardless, Petitioner is incorrect that combining the two references “would have involved nothing more than the simple substitution of Chowdhuri’s input tables for Pike’s input files, with predictable results.” Petition at 19. Indeed, if Chowdhuri’s customer and order tables were substituted for Pike’s input files, the output would simply be an amalgamation of entries with no coherent merger—the exact problem the ’610 patent aims to solve. And in fact, there is no coherent way in which Pike could be combined with Chowdhuri in the first place because the two references teach conflicting mechanisms.

Pike teaches traditional MapReduce, where data is first passed to “map processes” which are configured “to perform map functions and to execute an

application-specific map operator." Ex. 1002 at 3:59-61. Mapping the input data forms "intermediate data" in the form of key/value pairs. *Id*. at 7:29-58. The key/value pairs are passed to "reduce processes," which are configured "to perform reduce functions and to execute an application-specific reduce operator" and reduce the intermediate data based on common keys generated by the mapping. *Id*. at 3:61-64. The map phase and the reduce phase are each distributed "across multiple different machines on a distributed network." *Id*. at 3:40-43. This is very different from Chowdhuri.

As an initial matter, Petitioner is wrong that Chowdhuri's scanners are equivalent to map functions. *See*, *e.g.*, Petition at 20. Chowdhuri's "scanning" functions are never disclosed as performing any sort of "mapping"; instead, as recognized by Petitioner, Chowdhuri's scanning functions simply scan "rows from a table on disk" to be processed. Ex. 1003, ¶ 61; *see also* Petition at 26 ("each scanning iterator operates on its own thread and scans portions of the customer and order tables…").[14] Thus, what Chowdhuri actually teaches is that data from tables is

---

[14] That Chowdhuri refers to the scanned-in data as "intermediate results" does not mean that the scan functions "mapped" anything, and the only correspondence

simply scanned in (not "mapped" in any way) by scanning functions, such as from a disk, and scanned data from all tables is redistributed to a collection of "hash join" functions. Ex. 1003, ¶ 135; FIG. 3B. The "hash join" functions in Chowdhuri are disclosed as having both a "build operation" and a "probe operation"—this means that Chowdhuri's "hash join" functions create hash maps in the "build" phase and then join data on entries in the hash maps in the "probe" phase. *Id*., ¶ 130; *see generally* Ex. 2009.[15] Thus, while Chowdhuri and MapReduce both consolidate data, the two disclosures work very differently—Chowdhuri through hash join functions, and MapReduce through phases of distributed map and reduce functions.

There is thus no workable way to combine Pike with Chowdhuri because, contrary to Petitioner's argument, "Chowdhuri's scan, hashjoin, GroupBy, and sort

---

between Chowdhuri's "intermediate results" and Pike's "intermediate data" is the word "intermediate" itself.

[15] This emphasizes that Chowdhuri's scanners are not map functions, because the "build operation" of Chowdhuri's "hash join" function is what actually generates a hash map. Chowdhuri's scanning functions, on the other hand, simply scan in data to be processed. Thus, Chowdhuri's hash join functions are the only functions of Chowdhuri's disclosure that could possibly be likened to "map functions."

iterators" simply do not "correspond to" the map and reduce functions in Pike or the '610 patent. Petition at 20. If anything, Chowdhuri's hash join functions actually supplant *both* Pike's map functions and reduce functions, consolidating data but in a different way. This illustrates why Petitioner is wrong that POSITA would be motivated "to map and merge Chowdhuri's tables using Pike's distributed system." *Id*. Chowdhuri already teaches a way to consolidate data, i.e., by distributing collections of input data to singular hash join functions that accomplish the merger of tables without separate and distributed "map" and "reduce" functions. And Pike (which does not disclose merging tables at all) teaches a different mechanism, i.e., feeding input files to mapping functions to generate intermediate data, and then transmitting the intermediate data to be reduced. These teachings are mutually exclusive and cannot be combined as Petitioner suggests.

2. *Chowdhuri Does not Disclose Delineating "Data Groups" Prior to Partitioning.*[16]

Just like Pike, Chowdhuri also fails to disclose or render obvious these limitations. Again, Chowdhuri does not disclose "data groups." Petitioner also fails to explain what the "keys" and "values" are in either of the customer or order tables that Petitioner alleges constitute "data groups" and has thus failed to show that

---

[16] The same limitations relevant to Section VIII.A.2 are relevant here.

Chowdhuri discloses "key-value pairs." And again, as discussed above, Petitioner is wrong the combination of Pike and Chowdhuri "would have been nothing more than combining known elements in the prior art to yield the predictable result of efficiently processing Chowdhuri's input tables by partitioning each table into a plurality of partitions and processing them in a distributed, fault-tolerant system." Petition at 23-24. Chowdhuri's hash join functions serve similar purposes as Pike's mapping and reducing functions, so to combine the two references, Pike's entire MapReduce architecture would have to gutted and replaced with a collection of hash join functions, rendering impotent the entire parallelized architecture of Pike. In short, no combination of the two references can arrive at the challenged claims, meaning that there is no motivation to combine Pike with Chowdhuri.

3. *Chowdhuri Does not Disclose "Providing Each Data Partition to a Selected One of a Plurality of Mapping Functions."*[17]

Petitioner's combination of Pike and Chowdhuri fails to render any independent claim obvious because Chowdhuri does not disclose, and actually

---

[17] The limitations relevant to this argument are: [1d], [17c], [33d], and [40d] ("providing each data partition to a selected one of a plurality of mapping functions that are each user-configurable"); and [33h] and [40h] ("providing each data partition to a selected one of a plurality of mapping functions").

teaches away from, this claimed feature. As discussed above, instead of delineated "map" and "reduce" phases, Chowdhuri teaches a single hash join phase that builds a hash map and joins on entries within the hash map. *See* Ex. 1003, ¶ 130, FIG. 3; *see also*, *generally*, Ex. 2009. The hash join receives data from both the customer table and the order table. *See* FIG. 3B. Again, Chowdhuri's hash join function is the only function in Chowdhuri that could possibly be likened to a "map function." And Chowdhuri teaches providing pieces of each data table to every hash join function, not "providing each data partition to a selected one of a plurality of map functions."

This emphasizes that Chowdhuri and Pike cannot be combined to arrive at the present claims. Chowdhuri explicitly requires different functionality, and passing a partition from just one input table to a hash join function in Chowdhuri would cause Chowdhuri to be totally inoperable—no joinder of the data tables could be achieved, because the hash join function would only have data from one table. This is fatal to Petitioner's argument.

4. Chowdhuri Does not Disclose "Intermediate Data" that Is "Identifiable" to a "Data Group."[18]

Just like Pike, Chowdhuri also fails to disclose this feature. As an initial matter, Chowdhuri does not disclose generating "intermediate data" via "map

---

[18] The same limitations relevant to Section VIII.A.3 are relevant here.

functions" at all. Again, the scanning functions in Chowdhuri (which Petitioner wrongly likens to "map functions") do not "map" anything and just scan in "rows from a table on disk" to be processed. Ex. 1003, ¶ 61; *see also* Petition at 26. The results of the scanning functions thus cannot be considered "intermediate data" within the meaning of the '610 patent—indeed, the challenged claims require the "intermediate data" be output from map functions.

Confusingly, Petitioner claims that Chowdhuri's "o" and "c" from Chowdhuri's example query[19] are "data group" identifiers and "the results of Chowdhuri's scan iterator (e.g., map function) are identifiable to its input tables because the results are logically merged by customer state and customer order based on different input data to yield orders by state." Petition at 34. But this is nonsensical—Petitioner claims that the "o" and "c" are identifiers, but then does not allege that the "o" and "c" "identifiers" actually identify the purported "intermediate data" output from the scan iterators as belonging to any particular table (claimed "data groups") during data processing. That is because Chowdhuri does not teach this. To the contrary, standard SQL notation dictates that "customer c, order o" simply signifies that the customer table is being represented by the variable "c", and

---

[19] The example query includes "from customer c, order o." Ex. 1003, ¶ 126.

the order table is being represented by the variable "o", *within the query itself*, not within the scanners or hash join functions. Indeed, Chowdhuri never discloses that "o" and "c" are passed as identifiers to any functions that process the data in Chowdhuri, so "o" and "c" cannot be "mechanisms for identifying data" from "data groups" and do not make "intermediate data" "identifiable" to any given "data group" as required by the claims. And that the selected rows from Chowdhuri's tables are "logically merged" says nothing of *how* that merger is accomplished, much less that any sort of "a mechanism for identifying data" from a particular "data group" is used for accomplishing such merger.

Furthermore, an understanding of Chowdhuri's methodology emphasizes why Chowdhuri does not disclose intermediate data that is identifiable to a data group—Chowdhuri does not need a "mechanism" to identify which parent data group intermediate data came from because Chowdhuri accomplishes data consolidation *via the same function*, i.e., the hash join. This is different from the '610 patent, which employs "data groups" that specialized map and reduce functions can use to delineate between data (including "intermediate data") that originated from one group as compared to another to properly distribute and merge the data across the parallelized architecture. Indeed, in Chowdhuri, the data consolidation takes place in a self-contained function that is working with portions of *the original input data* after the input data has already been distributed. Chowdhuri's hash joins do not work

with data that has been transformed into "intermediate data" containing lists of key-value pairs, and the output from Chowdhuri's hash joins will not make its way to another phase of functions that will need to be able to identify what the original input data was. Put another way, Chowdhuri does need to maintain an identification of hash-mapped data as it moves to other function phases, because Chowdhuri avoids movement of hash-mapped data to other function phases entirely. In short, Chowdhuri does not disclose "intermediate data" that is "identifiable" to a parent "data group" because Chowdhuri has no such "intermediate data" and needs no such identifier.

5. *Chowdhuri Does not Disclose "Data Groups" with Different Schemas.*[20]

Petitioner's reliance on Chowdhuri to fill the holes in Pike with respect to these limitations is misplaced. Just like Pike, Chowdhuri never uses the term "schema," and further does not disclose "data groups" having different schemas. Petitioner alleges that the order and customer tables in Chowdhuri are "data groups" and that these tables have different schema. *See id*. at 40. While the order and customer tables in Chowdhuri may have different schema, they are not "data groups"—again, Chowdhuri discloses no "mechanism for identifying data" from

---

[20] The same limitations relevant to Section VIII.A.4 are relevant here.

these tables. And as discussed above, there is no motivation to combine Pike with Chowdhuri, because the references teach entirely different and conflicting mechanisms for parallel processing.

Further, with respect to Claims 33 and 40, Chowdhuri also cannot be said to disclose that an "output data set" from a reduce function "has a different schema than the first and second schema" of the input sets of key-value pairs, because Chowdhuri does not discuss "schema" at all, disclose what the schemas of the input sets are, or disclose what the schemas of the output sets are.

6. *Chowdhuri Does not Disclose a Key in Common Between the (1) "Data Groups"/Schema of the Input Data, and (2) the Intermediate Data.*[21]

Petitioner's reliance on Chowdhuri for teaching these limitations is misplaced. Again, Chowdhuri does not disclose "intermediate data" at all and thus cannot disclose this feature. And Petitioner fails, in both the Petition and its expert declaration, to allege any motivation to combine Chowdhuri with Pike to achieve the claim limitations. Indeed, Petitioner never explains why a POSITA would look to Chowdhuri for this limitation. *See KSR Int'l Co.*, 550 U.S. at 418 (explaining that the relevant question is whether there was an "apparent reason to combine the known

---

[21] The same limitations relevant to Section VIII.A.5 are relevant here.

elements in the fashion claimed by the patent at issue"); *see also LG Elecs., Inc. v. Cellular Commc'ns Equip. LLC*, IPR2016-00197, Paper 7 at 8 (PTAB April 29, 2016). Petitioner cannot explain this, because Chowdhuri offers no such teaching; to the contrary, Chowdhuri avoids making such "intermediate data" in the first place by performing data consolidation all within singular hash join functions. Petitioner's argument on this point is thus facially deficient.

7. *Chowdhuri Does not Disclose Processing Intermediate Data According to its "Data Group," or Merging the Intermediate Data Based on a Key in Common Between the Intermediate Data and the "Data Groups."*[22]

Petitioner next points to Chowdhuri as disclosing this element, alleging that "Chowdhuri renders this feature obvious by disclosing joining, grouping, and sorting by customer and order 'state.'" Petition at 49. This is wrong. Again, Chowdhuri does not disclose "intermediate data" at all and thus cannot teach this limitation—indeed, the claims require "reducing the *intermediate data* … so as to result in a merging of the corresponding different *intermediate data* based on the key in common," but Chowdhuri teaches performing hash joins on portions of the *original input data*, not on any "intermediate data" within the meaning of the '610 patent. And that Chowdhuri's hash join "has the notion of which data set to use for the build operation

---

[22] The same limitations relevant to Section VIII.A.6 are relevant here.

and what to use for the probe operation, normally designated by the left and the right operands respectively" does nothing to salvage Petitioner's argument. Petitioner argued previously that the "o" and "c" in Chowdhuri are purported "mechanisms for identifying data" from each table "through the scan, join, and group processes," but Petitioner cannot point to a single passage from Chowdhuri that teaches that the hash join function employs the "o" and "c" in this manner. Petition at 34. In fact, Chowdhuri discloses something different, namely, that the positioning of the operands within the function itself, not the "identity" of the original input data, guides the hash join function. *See* Ex. 1003, ¶ 130.

IX. CONCLUSION

For the foregoing reasons, Patent Owner respectfully requests that the Board decline to institute *inter partes* review of the '610 patent.

Dated: March 29, 2024

By: */s/Carder W. Brooks*

Carder W. Brooks
Registration No. 75,456
NELSON BUMGARDNER
CONROY P.C.
3131 W. 7th St.
Fort Worth, TX 76107
Telephone: (817) 806-3814
Email: carder@nelbum.com

Attorney for Patent Owner

CERTIFICATE OF COMPLIANCE WITH WORD COUNT

Pursuant to 37 C.F.R. § 42.24(d), I certify that this Preliminary Response complies with the type-volume limits of 37 C.F.R. § 42.24(b)(1) because it contains 13,966 words, excluding the parts of this Patent Owner's Preliminary Response that are exempted by 37 C.F.R. § 42.24(a), according to the word processing system used to prepare this Patent Owner's Preliminary Response.

Dated: March 29, 2024

By: */s/Carder W. Brooks*

Carder W. Brooks
Registration No. 75,456
NELSON BUMGARDNER
CONROY P.C.
3131 W. 7th St.
Fort Worth, TX 76107
Telephone: (817) 806-3814
Email: carder@nelbum.com

Attorney for Patent Owner

CERTIFICATE OF SERVICE

The undersigned certifies that the foregoing Patent Owner's Preliminary Response was served on March 29, 2024, to Lead and Back-up Counsel for Petitioner at the service address provided in Petitioner's Mandatory Notices:

Email: jeffrey.berkowitz@finnegan.com

daniel.tucker@finnegan.com

alexander.boyer@finnegan.com

taylor.stark@finnegan.com

Dated: March 29, 2024

By: */s/Carder W. Brooks*

Carder W. Brooks
Registration No. 75,456
NELSON BUMGARDNER
CONROY P.C.
3131 W. 7th St.
Fort Worth, TX 76107
Telephone: (817) 806-3814
Email: carder@nelbum.com

Attorney for Patent Owner